Terms used **bacon david compiler optimization** Found **29,031** of **185,178**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Compiler transformations for high-performance computing
David F. Bacon, Susan L. Graham, Oliver J. Sharp
December 1994 **ACM Computing Surveys (CSUR)**, Volume 26 Issue 4
Publisher: ACM Press
Full text available: pdf(6.32 MB) Additional Information: full citation, abstract, references, citings, index terms, review

In the last three decades a large number of compiler transformations for optimizing programs have been implemented. Most optimizations for uniprocessors reduce the number of instructions executed by the program using transformations based on the analysis of scalar quantities and data-flow techniques. In contrast, optimizations for high-performance superscalar, vector, and parallel processors maximize parallelism and memory locality with transformations that rely on tracking the properties o ...

Keywords: compilation, dependence analysis, locality, multiprocessors, optimization, parallelism, superscalar processors, vectorization

2 Design, implementation, and evaluation of optimizations in a just-in-time compiler
Kazuaki Ishizaki, Motohiro Kawahito, Toshiaki Yasue, Mikio Takeuchi, Takeshi Ogasawara, Toshio Suganuma, Tamiya Onodera, Hideaki Komatsu, Toshio Nakatani
June 1999 **Proceedings of the ACM 1999 conference on Java Grande**
Publisher: ACM Press
Full text available: pdf(1.09 MB) Additional Information: full citation, references, citings, index terms

3 On increasing architecture awareness in program optimizations to bridge the gap between peak and sustained processor performance: matrix-multiply revisited
David Parello, Olivier Temam, Jean-Marie Verdun
November 2002 **Proceedings of the 2002 ACM/IEEE conference on Supercomputing**
Publisher: IEEE Computer Society Press
Full text available: pdf(263.32 KB) Additional Information: full citation, abstract, references, citings, index terms

As the complexity of processor architectures increases, there is a widening gap between peak processor performance and sustained processor performance so that programs now tend to exploit only a fraction of available performance. While there is a tremendous

amount of literature on program optimizations, compiler optimizations lack efficiency because they are plagued by three flaws: (1) they often implicitly use simplified, if not simplistic, models of processor architecture, (2) they usually foc ...

4 Code optimization - I: Compiler optimization-space exploration

Spyridon Triantafyllis, Manish Vachharajani, Neil Vachharajani, David I. August

March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization CGO '03**

Publisher: IEEE Computer Society

Full text available: pdf(1.19 MB) Additional Information: full citation, abstract, references, citings, index terms

To meet the demands of modern architectures, optimizing compilers must incorporate an ever larger number of increasingly complex transformation algorithms. Since code transformations may often degrade performance or interfere with subsequent transformations, compilers employ predictive heuristics to guide optimizations by predicting their effects a priori. Unfortunately, the unpredictability of optimization interaction and the irregularity of today's wide-issue machines severely limit the accura ...

5 Optimized unrolling of nested loops

Vivek Sarkar

May 2000 **Proceedings of the 14th international conference on Supercomputing**

Publisher: ACM Press

Full text available: pdf(1.10 MB) Additional Information: full citation, abstract, references, citings, index terms

In this paper, we address the problems of automatically selecting unroll factors for perfectly nested loops, and generating compact code for the selected unroll factors. Compared to past work, the contributions of our work include a) a more detailed cost model that includes ILP and 1-cache considerations, b) a new code generation algorithm for unrolling nested loops that generates more compact code (with fewer remainder loops) than the unroll-and-jam transf ...

6 A study of devirtualization techniques for a Java Just-In-Time compiler

Kazuaki Ishizaki, Motohiro Kawahito, Toshiaki Yasue, Hideaki Komatsu, Toshio Nakatani

October 2000 **ACM SIGPLAN Notices , Proceedings of the 15th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '00**, Volume 35 Issue 10

Publisher: ACM Press

Full text available: pdf(225.89 KB) Additional Information: full citation, abstract, references, citings, index terms

Many devirtualization techniques have been proposed to reduce the runtime overhead of dynamic method calls for various object-oriented languages, however, most of them are less effective or cannot be applied for Java in a straightforward manner. This is partly because Java is a statically-typed language and thus transforming a dynamic call to a static one does not make a tangible performance gain (owing to the low overhead of accessing the method table) unless it is inlined, and partly because t ...

7 Formal semantics and static analysis: Compiler optimizations for nondeferred reference: counting garbage collection

Pramod G. Joisha

June 2006 **Proceedings of the 2006 international symposium on Memory management ISMM '06**

Publisher: ACM Press

Full text available: pdf(220.00 KB) Additional Information: full citation, abstract, references, index terms

Reference counting is a well-known technique for automatic memory management, offering unique advantages over other forms of garbage collection. However, on account of the high costs associated with the maintenance of up-to-date tallies of references from the stack, deferred variants are typically used in modern implementations. This partially sacrifices some of the benefits of non-deferred reference-counting (RC) garbage collection, like the immediate reclamation of garbage and short collector ...

Keywords: reference counting, static analyses

8 Fast interprocedural class analysis

Greg DeFouw, David Grove, Craig Chambers

January 1998 **Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**

Publisher: ACM Press

Full text available: pdf(2.03 MB) Additional Information: full citation, references, citings, index terms

9 Vortex: an optimizing compiler for object-oriented languages

Jeffrey Dean, Greg DeFouw, David Grove, Vassily Litvinov, Craig Chambers

October 1996 **ACM SIGPLAN Notices , Proceedings of the 11th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '96**, Volume 31 Issue 10

Publisher: ACM Press

Full text available: pdf(2.45 MB) Additional Information: full citation, abstract, references, citings, index terms

Previously, techniques such as class hierarchy analysis and profile-guided receiver class prediction have been demonstrated to greatly improve the performance of applications written in pure object-oriented languages, but the degree to which these results are transferable to applications written in hybrid languages has been unclear. In part to answer this question, we have developed the Vortex compiler infrastructure, a language-independent optimizing compiler for object-oriented languages, with ...

10 Power optimization using divide-and-conquer techniques for minimization of the number of operations

Inki Hong, Miodrag Potkonjak, Ramesh Karri

October 1999 **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, Volume 4 Issue 4

Publisher: ACM Press

Full text available: pdf(278.45 KB) Additional Information: full citation, abstract, references, index terms

We introduce an approach for power optimization using a set of compilation and architectural techniques. The key technical innovation is a novel divide-and-conquer compilation technique to minimize the number of operations for general computations. Our technique optimizes not only a significantly wider set of computations than the previously published techniques, but also outperforms (or performs at least as well as other techniques) on all examples. Along the architectural dimension, we in ...

Keywords: code generation, transformations

11 Predicting the impact of optimizations for embedded systems

Min Zhao, Bruce Childers, Mary Lou Soffa

June 2003 **ACM SIGPLAN Notices , Proceedings of the 2003 ACM SIGPLAN conference on Language, compiler, and tool for embedded systems LCTES '03**, Volume

38 Issue 7
Publisher: ACM Press

Full text available: pdf(281.53 KB) Additional Information: full citation, abstract, references, citings, index terms

When applying optimizations, a number of decisions are made using fixed strategies, such as always applying an optimization if it is applicable, applying optimizations in a fixed order and assuming a fixed configuration for optimizations such as tile size and loop unrolling factor. While it is widely recognized that these fixed strategies may not be the most appropriate for producing high quality code, especially for embedded systems, there are no general and automatic strategies that do otherwi ...

Keywords: code models, embedded systems, loop optimizations, optimization models, optimizing compilers, prediction, resource models

12 Diverse topics: Field level analysis for heap space optimization in embedded java environments

Guangyu Chen, Mahmut Kandemir, N. Vijaykrishnan, Mary Janie Irwin
October 2004 **Proceedings of the 4th international symposium on Memory management**
Publisher: ACM Press
Full text available: pdf(1.67 MB) Additional Information: full citation, abstract, references, index terms

Memory constraint presents one of the critical challenges for embedded software writers. While circuit-level solutions based on cramming as many bits as possible into the smallest area possible are certainly important, memory-conscious software can bring much higher benefits. Focusing on an embedded Java-based environment, this paper studies potential benefits and challenges when heap memory is managed at a field granularity instead of object. This paper discusses these benefits and challenge ...

Keywords: garbage collection, java virtual machine

13 Exploiting non-uniform reuse for cache optimization

Claudia Leopold
March 2001 **Proceedings of the 2001 ACM symposium on Applied computing**
Publisher: ACM Press
Full text available: pdf(110.43 KB) Additional Information: full citation, references, index terms

Keywords: caching, data locality, program restructuring

14 Using types to analyze and optimize object-oriented programs

Amer Diwan, Kathryn S. McKinley, J. Eliot B. Moss
January 2001 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 23 Issue 1
Publisher: ACM Press

Full text available: pdf(414.51 KB) Additional Information: full citation, abstract, references, citings, index terms

Object-oriented programming languages provide many software engineering benefits, but these often come at a performance cost. Object-oriented programs make extensive use of method invocations and pointer dereferences, both of which are potentially costly on modern machines. We show how to use types to produce effective, yet simple, techniques that reduce the costs of these features in Modula-3, a statically typed, object-oriented

language. Our compiler performs type-based alias analysis to ...

Keywords: alias analysis, classes and objects, method invocation, object orientation, polymorphism, redundancy elimination

15 Ulterior reference counting: fast garbage collection without a long wait

Stephen M. Blackburn, Kathryn S. McKinley
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications OOPSLA '03**, Volume 38 Issue 11
Publisher: ACM Press

Full text available: pdf(218.61 KB) Additional Information: full citation, abstract, references, citings, index terms

General purpose garbage collectors have yet to combine short pause times with high throughput. For example, generational collectors can achieve high throughput. They have modest average pause times, but occasionally collect the whole heap and consequently incur long pauses. At the other extreme, concurrent collectors, including reference counting, attain short pause times but with significant performance penalties. This paper introduces a new hybrid collector that combines copying generational c ...

Keywords: Java, copying, generational hybrid, reference counting, ulterior reference counting

16 Technical correspondence: What can we gain by unfolding loops?

Litong Song, Krishna Kavi
February 2004 **ACM SIGPLAN Notices**, Volume 39 Issue 2
Publisher: ACM Press
Full text available: pdf(989.18 KB) Additional Information: full citation, abstract, references

Loops in programs are the source of many optimizations for improving program performance, particularly on modern high-performance architectures as well as vector and multithreaded systems. Techniques such as loop invariant code motion, loop unrolling and loop peeling have demonstrated their utility in compiler optimizations. However, many of these techniques can only be used in very limited cases when the loops are "well-structured" and easy to analyze. For instance, loop invariant code motion w ...

Keywords: loop peeling, loop quasi invariant code motion, loop unrolling, quasi-index variable, quasi-invariant variable

17 Simple and effective analysis of statically-typed object-oriented programs

Amer Diwan, J. Eliot B. Moss, Kathryn S. McKinley
October 1996 **ACM SIGPLAN Notices , Proceedings of the 11th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '96**, Volume 31 Issue 10
Publisher: ACM Press

Full text available: pdf(1.70 MB) Additional Information: full citation, abstract, references, citings, index terms

To use modern hardware effectively, compilers need extensive control-flow information. Unfortunately, the frequent method invocations in object-oriented languages obscure control flow. In this paper, we describe and evaluate a range of analysis techniques to convert method invocations into direct calls for statically-typed object-oriented languages and thus improve control-flow information in object-oriented languages. We present

simple algorithms for *type hierarchy analysis, aggregate analys ...*

18 Compiler-directed page coloring for multiprocessors

Edouard Bugnion, Jennifer M. Anderson, Todd C. Mowry, Mendel Rosenblum, Monica S. Lam
September 1996 **ACM SIGPLAN Notices , ACM SIGOPS Operating Systems Review , Proceedings of the seventh international conference on Architectural support for programming languages and operating systems ASPLOS-VII**, Volume 31 , 30 Issue 9 , 5
Publisher: ACM Press

Full text available: pdf(1.37 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a new technique, *compiler-directed page coloring,* that eliminates conflict misses in multiprocessor applications. It enables applications to make better use of the increased aggregate cache size available in a multiprocessor. This technique uses the compiler's knowledge of the access patterns of the parallelized applications to direct the operating system's virtual memory page mapping strategy. We demonstrate that this technique can lead to significant performance impr ...

19 Memory aspects in system design: Loop fusion for memory space optimization

Antoine Fraboulet, Karen Kodary, Anne Mignotte
September 2001 **Proceedings of the 14th international symposium on Systems synthesis**
Publisher: ACM Press
Full text available: pdf(152.91 KB) Additional Information: full citation, abstract, references, index terms

Portable or embedded systems as well as submicronic technologies have made the power consumption criterium crucial. Memory is known to be extremely power consuming. Moreover multimedia applications are memory intensive applications. Therefore, we propose new techniques to optimize a behavioral description of multimedia applications before the hardware/software partitioning *(Codesign)*. These transformations are performed on "for" loops that constitute the main parts which handle the arrays ...

20 Array recovery and high-level transformations for DSP applications

Björn Franke, Michael O'boyle
May 2003 **ACM Transactions on Embedded Computing Systems (TECS)**, Volume 2 Issue 2
Publisher: ACM Press

Full text available: pdf(744.35 KB) Additional Information: full citation, abstract, references, citings, index terms

Efficient implementation of DSP applications is critical for many embedded systems. Optimizing compilers for application programs, written in C, largely focus on code generation and scheduling, which, with their growing maturity, are providing diminishing returns. As DSP applications typically make extensive use of pointer arithmetic, the alternative use of high-level, source-to-source, transformations has been largely ignored. This article develops an array recovery technique that automatically ...

Keywords: Pointer conversion, dataflow graphs, embedded processors, high-level transformations